Securely Taking on New Executable Software of Uncertain Provenance (STONESOUP) Program Overview

Software Assurance Forum
30 September 2010

W. Konrad Vesey
IARPA

STONESOUP Program

- Program kicked off September 2010
- BAA closed
- Performers:
  - GrammaTech
  - SAIC
  - Kestrel Institute
  - University of Illinois Urbana-Champagne
  - Columbia University
- Test and evaluation team is led by MITRE

STONESOUP Goals

- Provide end users/enterprises the capability to transform a software executable to create a safer version
- Automatically analyze, confine, and diversify software without regard to its provenance
- Address implementation defects, not design defects
- Address inadvertent defects, not malicious logic

Technical Approach

- Analysis
  - Must be fully automated
- Confinement
  - Render weaknesses identified by analysis unexploitable
- Diversification
  - Address residual risk to raise the cost or lower the impact of an attack on the software

Targeted Software

- Each proposal chose a class of software to target from among:
  - A type-safe language (Java or C#--no proposer chose C#)
  - A non-type-safe languages (C or C++)
  - Binary executables (x86 Windows or x86 Linux)

Targeted Weaknesses

- Each proposal chose six classes of weakness to target from among:
  1. Number handling
  2. Tainted data/Input Validation errors
  3. Error handling
  4. Resource drains
  5. Injections (SQL, command)
  6. Concurrency handling/Race conditions
  7. Buffer overflows/Memory safety violations (C, C++, binary only)
  8. Null pointer errors (C, C++, binary only)

Program Balance

- The STONESOUP Program has achieved good coverage over the space of possible targets:

	1	2	3	4	5	6	7	8
A	✓✓	✓✓	✓✓	✓	✓✓	✓		
B	✓✓	✓✓	✓✓		✓	✓	✓✓	✓
C	✓	✓✓	✓✓	✓✓	✓✓	✓	✓✓	✓✓

Program Metrics (1)

- STONESOUP metrics focus on soundness: identification and neutralization of nearly all vulnerabilities of a given class:
  - Phase 1 (18 mos): Solutions must find and render unexploitable 75% of the vulnerabilities in a suite of small (<10K SLOC) test programs
  - Phase 2 (12 mos): 90% of the phase 1 vulnerabilities and 80% of the vulnerabilities in a suite of medium-sized (~100K SLOC) test programs
  - Phase 3 (18 mos): 95% of the phase 1&2 vulnerabilities and 90% of the vulnerabilities in a suite of large (~500K SLOC) test programs
  - For each phase a small percentage of C, C++, or binary test programs may be rejected (not processed)

Program Metrics (2)

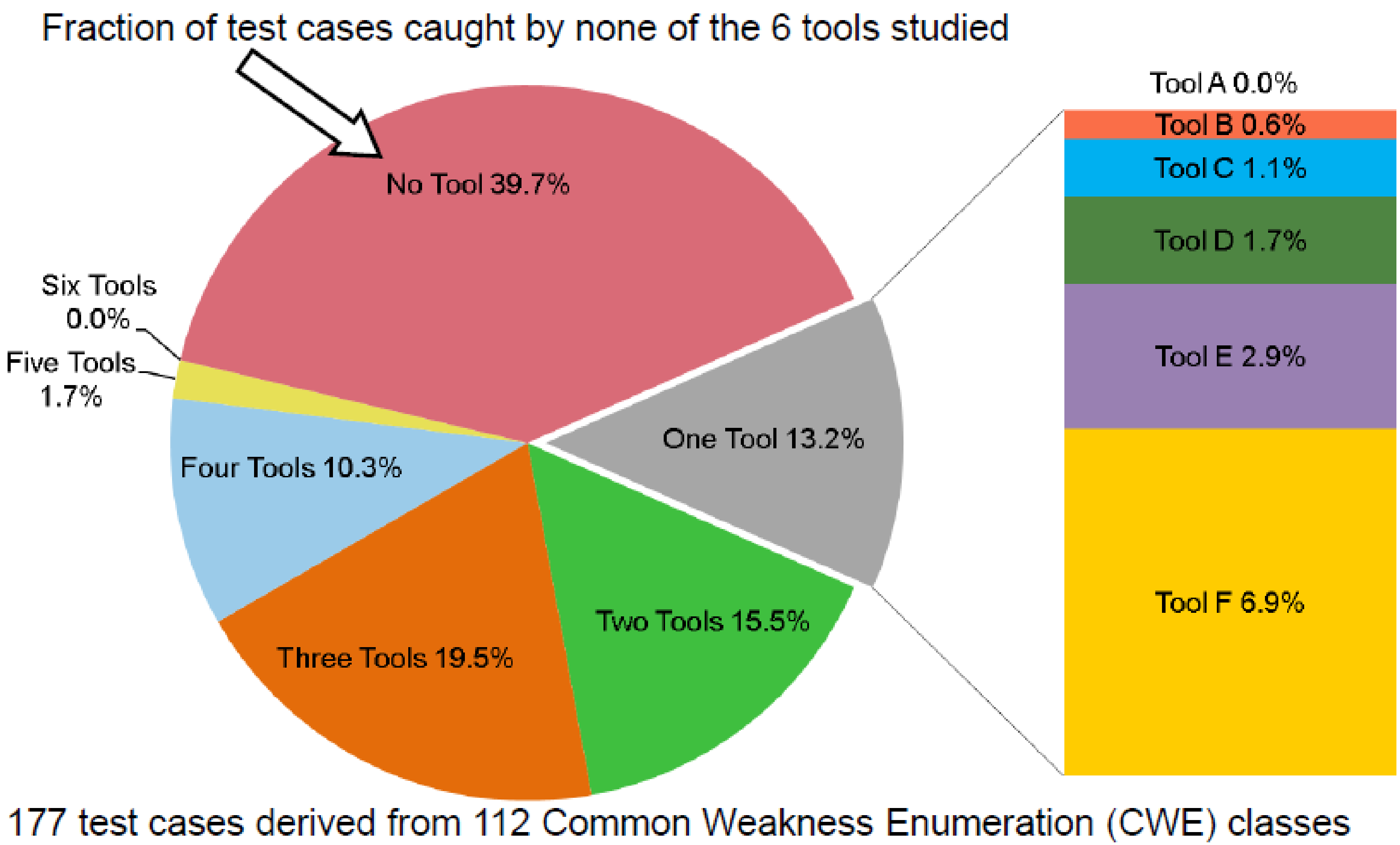
- Phase 3 includes
  - Measurement of the performance overhead of the processed software (no more than +10% increase in running time is sought)
  - Assessment of the additional work factor imposed on an attacker due to the diversification (measures to be determined)

Program Reality Check

- Test and Evaluation team has been tasked by Program Manager to develop a baseline solution using an integrated toolbox of commercial off the shelf products
  - Will be evaluated in conjunction with STONESOUP Test and Evaluation to provide context
  - Likely will not be fully automated/integrated: goal is to get reasonably close given resource constraints
  - Some custom generation of “rules” may be allowed, but not custom tool development

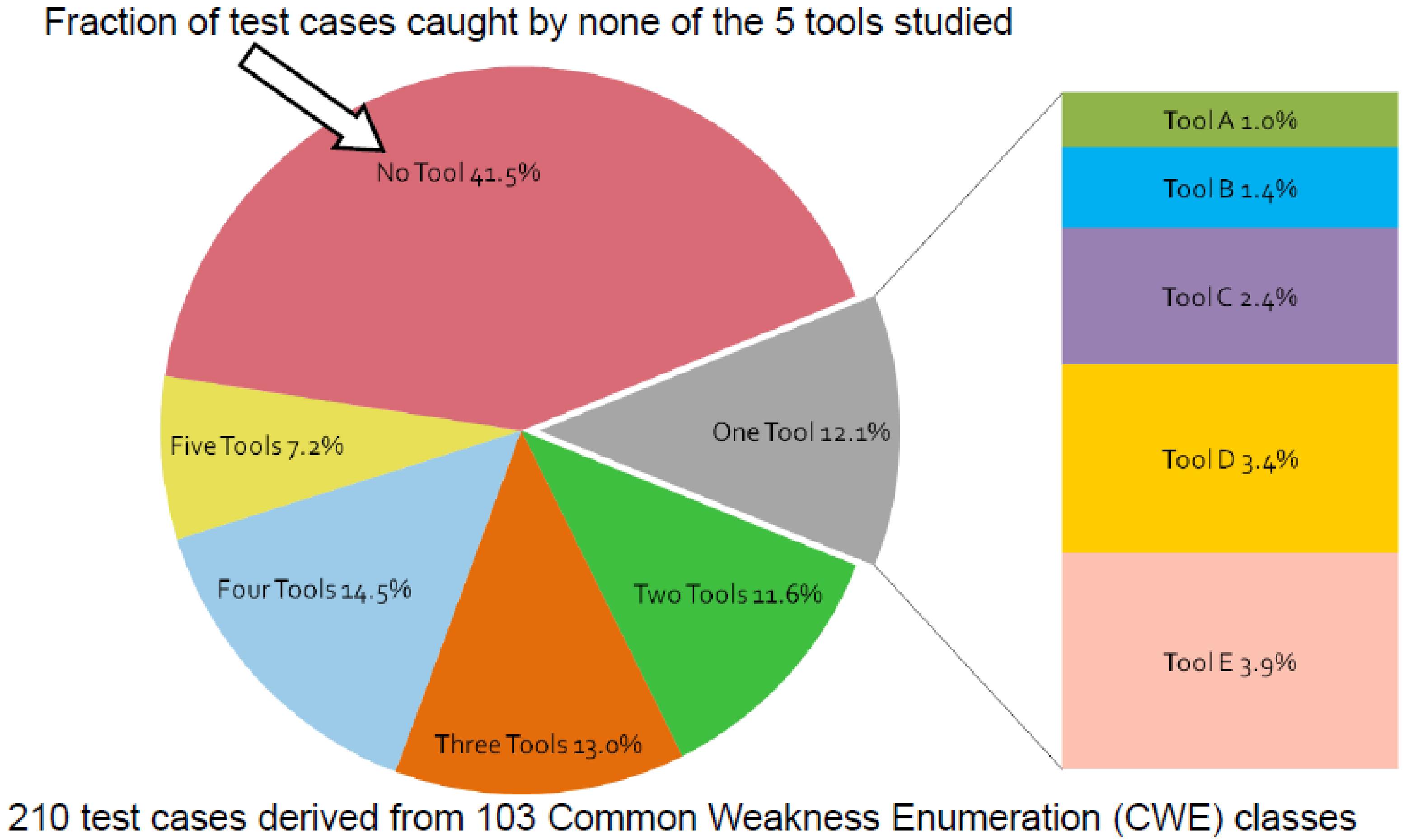
Analysis Advances Sought (1)

- Results of a 2009 study of Java tools:



Analysis Advances Sought (2)

- Results of a 2009 study of C/C++ tools:



Analysis Advances Sought (3)

- Current analysis techniques lead to an end-user decision
  - False positives must be identified by human review
  - False negatives are common
  - The end-user must select and apply countermeasures
- STONESOUP analysis must lead to automated response
  - Program metrics are designed to drive advances in both precision and soundness
    - False positives could result in unnecessary countermeasures, degrading performance
    - False negatives could result in failure to meet the test and evaluation targets

Confinement Advances Sought (1)

- STONESOUP confinement must be precise enough to limit performance overhead
- Confinement may be software or hardware based, but should use widely available commodity systems
- Automation requirements drive novel approaches to error response and recovery

Confinement Advances Sought (2)

- Weakness targeting requirements drive novel mixtures of confinement techniques
  - Precise/sound input filtering
  - Runtime inspection of program state
  - Resource virtualization
  - Adaptive code rewriting
- Diversification requirements drive exploration of variations within each confinement technique

Diversification Advances Sought (1)

- Current diversification techniques address generic attack elements
  - Program state data is more difficult to find or predict
  - Faulty program states are more difficult to replicate
  - Exploitation impact is more difficult to predict
- Current diversification is rarely driven by analysis
  - Vulnerabilities may simply be shuffled around
  - Uncertain where and how much diversification can be applied due to lack of data dependency information that analysis could provide

Diversification Advances Sought (2)

- STONESOUP seeks diversification techniques that protect against specific classes of weakness
  - Program requires effectiveness estimates for chosen diversification techniques against a targeted weakness class
- Diversification techniques can be mixed
  - Address space layout randomization + instruction set randomization + function call variants, etc.
  - Techniques can be adaptive
  - Each executing instance may be unique, and thus at most one instance can be exploited with each malicious input

Securely Taking on New Executable Software of Uncertain Provenance (STONESOUP) Program

W. Konrad Vesey
Program Manager
Office of Safe and Secure Operations
Intelligence Advanced Research Projects Activity
william.k.vesey@ugov.gov